HOW TO GIT SECURITY

LOCAL
considerations

Credentials
Be aware whether, where and how your credentials are stored. Prefer to log-in with keys instead of your password.

Gitignore
You can exclude specific files by listing the patterns in a file named `.gitignore` in the root of your repository:

```
$ cat > .gitignore
# Log File folder.
**/logs/
# Environment variables.
.env
# Temporary data.
data/
^D
```

Generate Keys and clone using SSH
Use SSH Keys to connect with servers.

```
$ ssh-keygen -t ed25519 -C "leo@isec.de"
Generating public/private ed25519 key pair.
...
The key fingerprint is:
SHA256:onZbNXrSrrScrfzrer7bi7DbsUrFvC515f0s21QwG7k Hi ho
```

Sign your commits
Verified

Signed commits allow to verify authenticity of your commits.

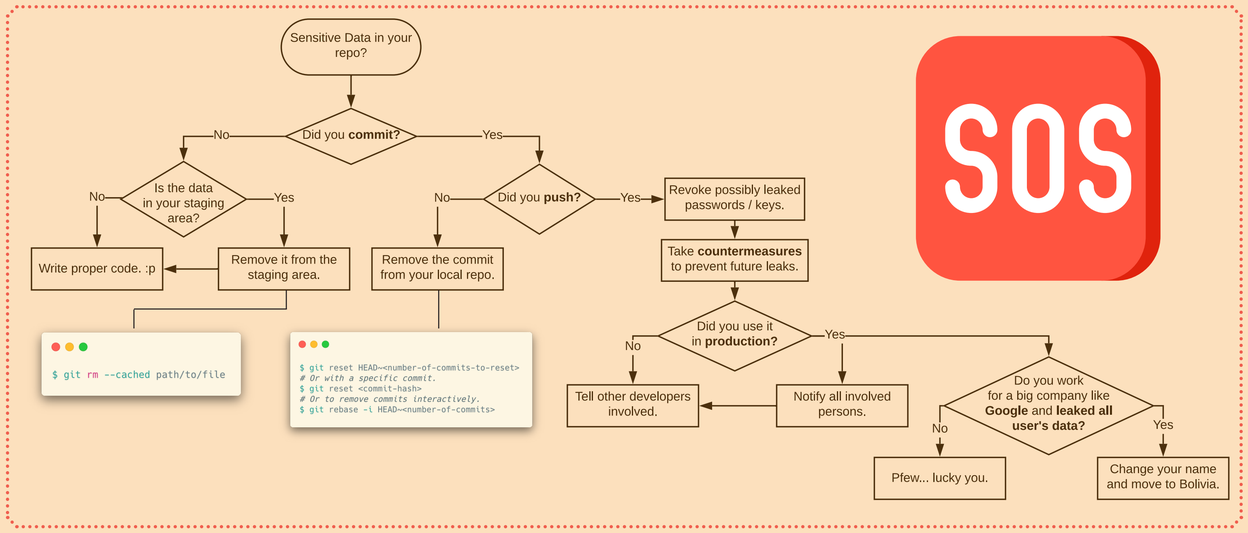
```
$ gpg --full-generate-key
  ...
$ gpg --list-secret-keys --keyid-format LONG your@mail.com
sec   rsa2048/37EAD8CA8C24F789 2020-01-30 [SC] [expires: 2022-01-29]
      385E670ED1F9BF887E6DABCA07EA64C78A746789
uid                 [ultimate] Evil Platypus <your@mail.com>
ssb   rsa2048/28134627A631BEA1 2020-01-30 [E] [expires: 2022-01-29]
$ gpg --armor --export 37EAD8CA8C24F789
# copy this to GitHub / GitLab
$ git config --global user.signingkey 37EAD8CA8C24F789
$ git config --global commit.gpgsign true
```

Use .env files
.env files allow you to set environment variables.
List them in a `VARIABLE=VALUE` format:

```
$ cat .env
EMAIL_SERVER='some.email.de'
EMAIL_PORT=25
USER='leo'
PASSWORD='somepasswordleowouldchooseforemails'
^D
```

```
#!/usr/bin/env python3
import os
from dotenv import load_dotenv

load_dotenv()
secret_password = os.getenv("SERVER_PASSWORD")
```



Software-Security
considerations

GIT was meant to be **decentralized!**

Hosting
Keep in mind that those **services** have been and could still be **vulnerable!**

Maybe **self-hosting** is an option!

Please remember that self-hosting has some **implications** like protecting physical access and backing up data etc. being **your** responsibility.

Authentication
Use two-factor authentication whenever possible. GitHub recommends to use a 2FA-App (e.g.: Authy, Google Authenticator) instead of SMS.

Be careful about SHA1
Git uses SHA1 to generate your commit IDs. Be careful that SHA1 no longer is considered secure and collisions can be crafted.

It's possible to exchange a commit and have the same ID!

Linus Torvalds didn't think about the algorithm being replacable. The team is working since 2014 on preparing the transition to **SHA256**.

USEFUL TOOLS

The following tools can help to identify potential leaks. Consider using them in pre-commit-hooks.

GitHub Dorks
Search GitHub for sensitive data like `filetype:.env`

TruffleHog
TruffleHog searches all branches and the whole commit history for high-entropy data and other possible leaks.

shhgit
shhgit is a website to monitor public repositories live for possible breaches via the GitHub Event API.

git-secrets
In `git-secrets` you can define your own rules for checking your code before committing.

GitGot
GitGot is a semi-automated, feedback-driven tool to empower users to rapidly search through troves of public data on GitHub for sensitive secrets.

Security Considerations Using git by Felix Gohla, Tim Garrels, Benito Buchheim